To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

HEW

HEW Code Generation (CodeGen)

Introduction

This Application Note gives an in-depth explanation on various setting of the project generation.

The generated files and start up flow of the MCU are explained.

An example of porting a C code from SLP to TINY is also illustrated.

Target Device

All

Contents

1. Overview

High-performance Embedded Workshop (HEW2.2) from Renesas is a flexible code development and debugging environment for applications targeted at Renesas microcontrollers. It provides an up-to-date “look and feel” with all of the features you would expect from a modern development environment.

The main features are:

- A configurable build engine that allows you to set-up compiler, assembler and linker options via in easy to use interface.
- An integrated text editor with user customizable syntax coloring to improve code readability.
- A configurable environment to run your own tools.
- An integrated debugger which allows you to build and debug in the same application.

2. Invoking the HEW

After the installation of the HEW, the installer creates a folder whose name “Renesas High-performance Embedded Workshop” in the “Program” folder of the “Start” menu of the Windows®. In the “Renesas High-performance Embedded Workshop” folder, shortcut of the HEW support files will be registered. The contents of the “Start” menu and its submenu maybe differ depending on your installation.

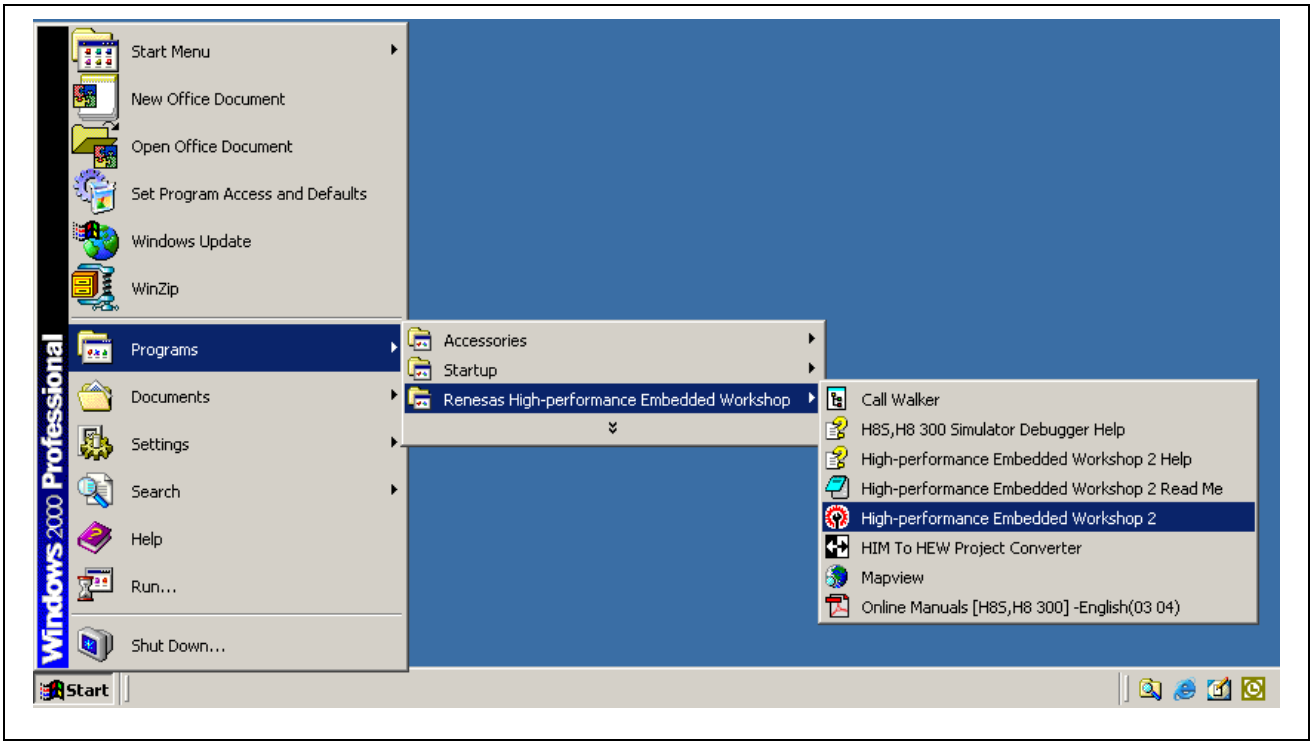

Figure 1 Invoking the HEW from the “Start” Menu

If you click “High-performance Embedded Workshop 2” from the menu, the HEW will be invoked and the “Welcome!” dialog box (Figure 2) will be displayed.

Alternatively, you can open the project promptly without the “Welcome!” dialog box if you change the setting via “Tools->Options”. *For details of this setting, refer to chapter 6, “Customizing the environment”, of the High-performance Embedded Workshop 2 User’s Manual.*

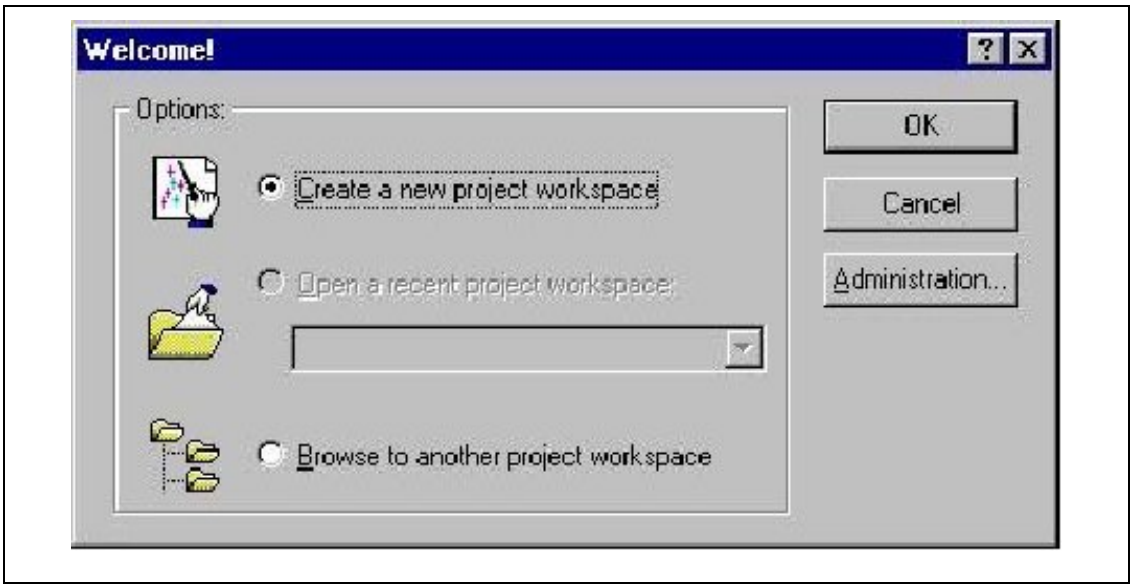

Figure 2 Welcome! Dialog Box

If you use the HEW for the first time or if you want to work on a new project, select the “Create a new project workspace” radio button and click “OK”.

If you want to work on an existing project, select the “Open a recent project workspace” or “Browse to another project workspace” radio button and click “OK”.

If you want to cancel the HEW, click “Cancel”.

If you want to control the component in the HEW, click on “Administration…”, which will invoked via “Tool->Administration…”. *For details of this setting, refer to chapter 5, “Tool Administration”, of the High-performance Embedded Workshop 2 User’s Manual.*

3. Creating a Project

3.1 Creating a New Workspace

When you have selected the “Create a new project workspace” radio button and clicked “OK” on the “Welcome!” dialog box, the “New Project Workspace” dialog box (Figure 3), which is used to create a new workspace and project, will be launched. You will specify a workspace name (When a new workspace is created, the project name is the same as the default), a CPU family, a project type, and so on, on this dialog box.

Enter the name of your workspace, “Tutorial”, for example, in the “Workspace Name” field, and the “Project Name” field will show “tutorial” and the “Directory” field will show “C:\Hew2\Tutorial”.

If you want to change the project name, enter a new project name manually in the “Project Name” field.

If you want to change the directory used for the new workspace, click the “Browse…” button and specify a directory, or enter a directory path manually in the [Directory] field.

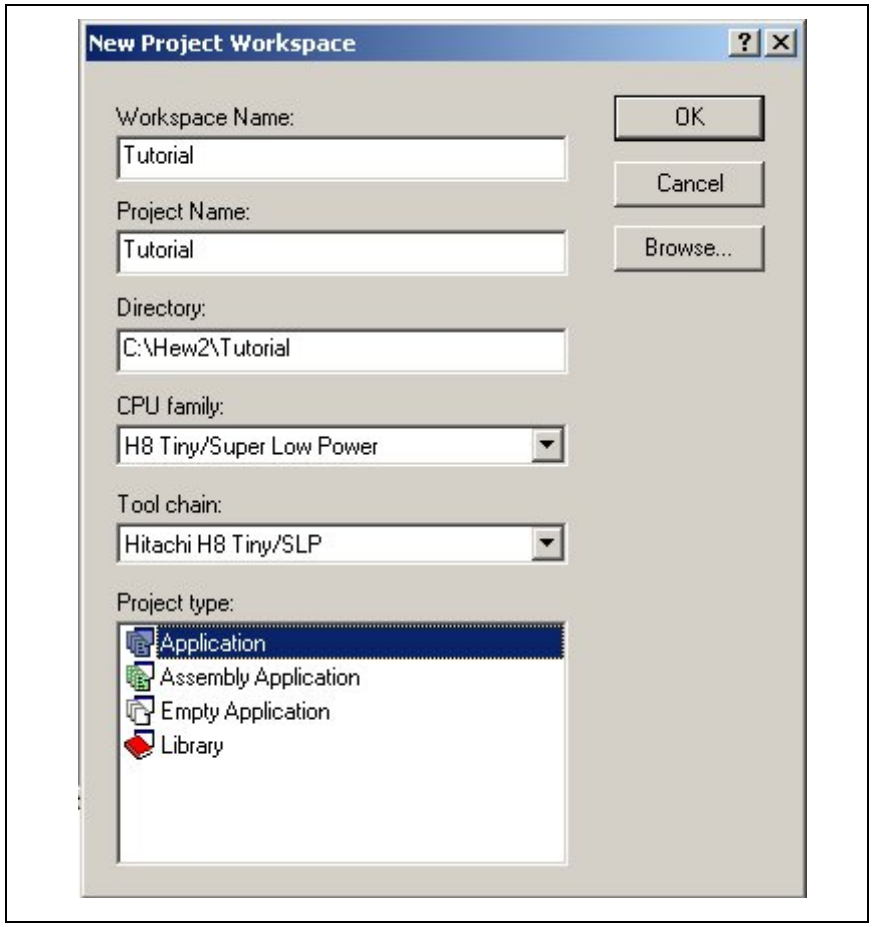

Figure 3 New Project Workspace Dialog Box

Select the CPU family from the drop-down list of “CPU family” combo box. If you are not sure the CPU family of the microcontroller, you can search from our website at “http://www.renesas.com/eng/products/mpumcu/index.html” or contact our sale representative for clarification.

The “Tool chain” combo box is depending on the installed toolchain which you selected earlier. HEW will automatically select the relevant tool chain based on the CPU family such as SLP, H8, TINY, SH etc.

Table 1 Item of "Project type"

"Project type:"	Description	Type of Files Generated *
Application	This project is used to create a program that includes the initial routine files written in C/C++ language.	"dbsct.c" "intprg.c" "resetprg.c" "sbrk.c" "<Workspace name>.c" "sbrk.h" "stacksct.h"
Assembly Application	This project is used to create a program that includes the initial routine files written in assembly language.	"vecttbl.src" "intprg.src" "resetprg.src" "<Workspace name>.src" "stacksct.src" "vect.inc"
Empty Application	This project is used to set up the tool chain.	No file is to be generated
Library	This project is used to create a library file.	No file is to be generated

**Please refer to 3.0 File Generated for detail function of each file.*

Note For this example, we selected Application as the "Project type" before proceed.

3.2 Selecting the Target CPU

When you click “OK” on the “New Project Workspace” dialog box, the project generator will be invoked. Start by selecting the CPU that you will be using. CPU types shown in the “CPU Type:” list are classified into the CPU series shown in the “CPU Series:” list.

The selected items in the “CPU Series:” list box and the “CPU Type:” list box specify the files to be generated. Select the CPU type of the program to be developed. If the CPU type which you want to select is not displayed in the “CPU Type:” list, select a CPU type with similar hardware specifications or select “Other”.

Clicking “Next>” moves to the next display. Clicking “<Back” moves to the previous display or the previous dialog box. Clicking “Finish” opens the “Summary” dialog box. Clicking “Cancel” returns the display to the “New Project Workspace” dialog box. “<Back”, “Next>”, “Finish”, and “Cancel” are common buttons of all the wizard dialog boxes.

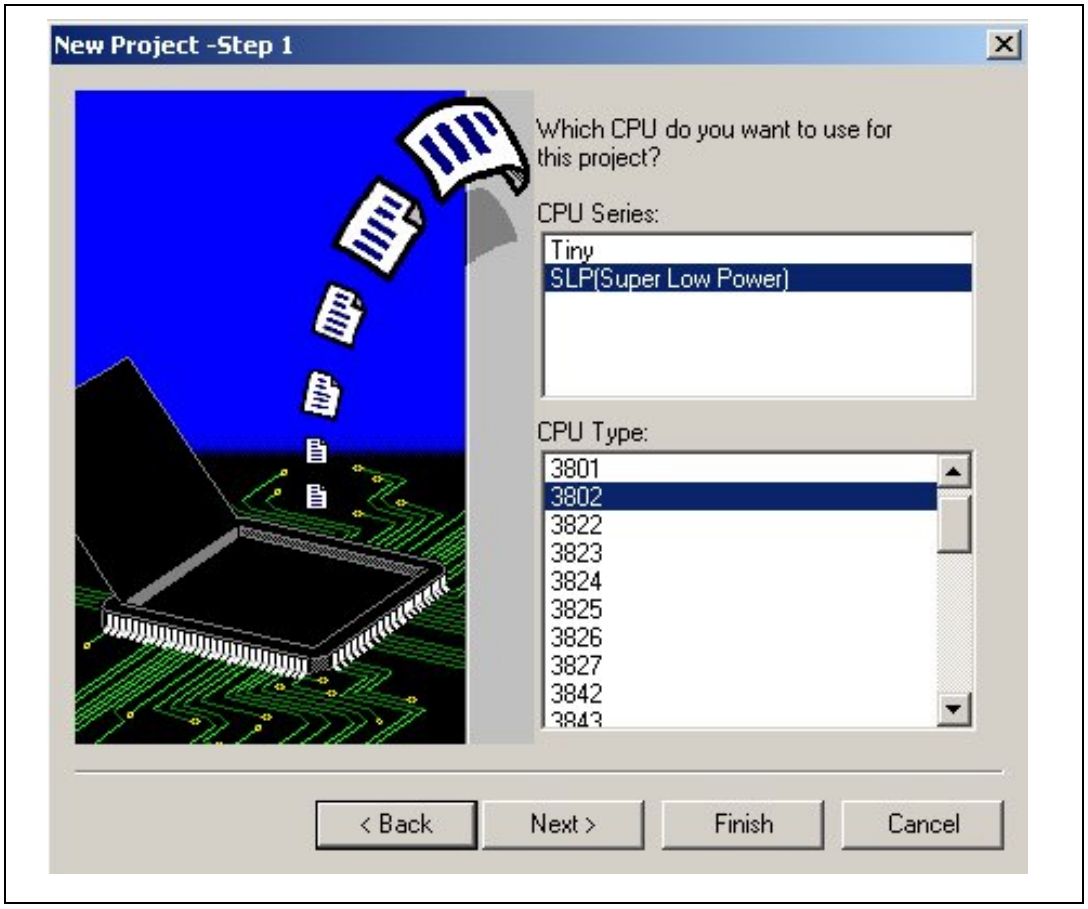

Figure 4 Selecting the Target CPU (Step 1)

Note: For this example, we selected SLP (Super Low Power) as CPU Series and 3802 as CPU Type before proceed.

3.3 Option Setting

There are two ways to approach “Option Setting” dialog box that are shown below.

1. When creating a new project:

Clicking the “Next>” button on the Step 1 screen opens the dialog box shown in Figure 5.

On this screen, you can specify the options common to all project files in Step 2. The specifiable items depend on the CPU selected in Step 1.

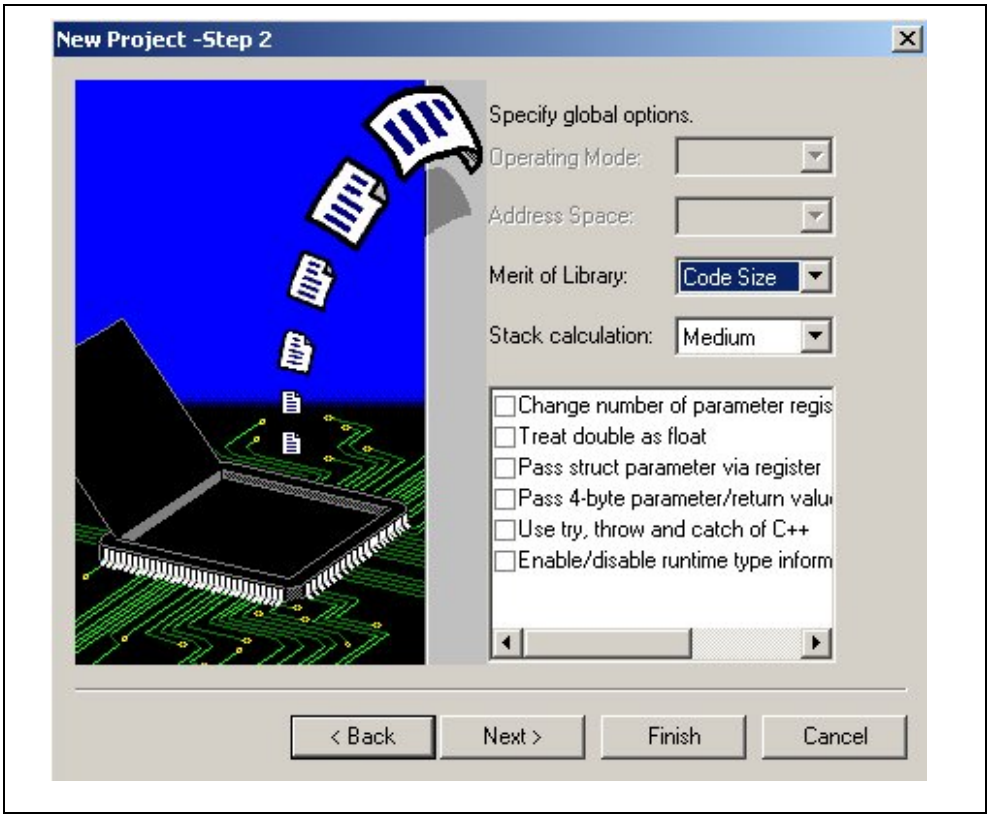

Figure 5 Option Setting (Step 2)

2. After a project had been built:

Above similar changes can be made even a project had been built. Under Menu Bar Section, click on [Options->Hitachi H8 Tiny/SLP Toolchain->CPU], as shown below in Figure 6.

Note: Above Toolchain type is dependent on the type of CPU Series being chosen.

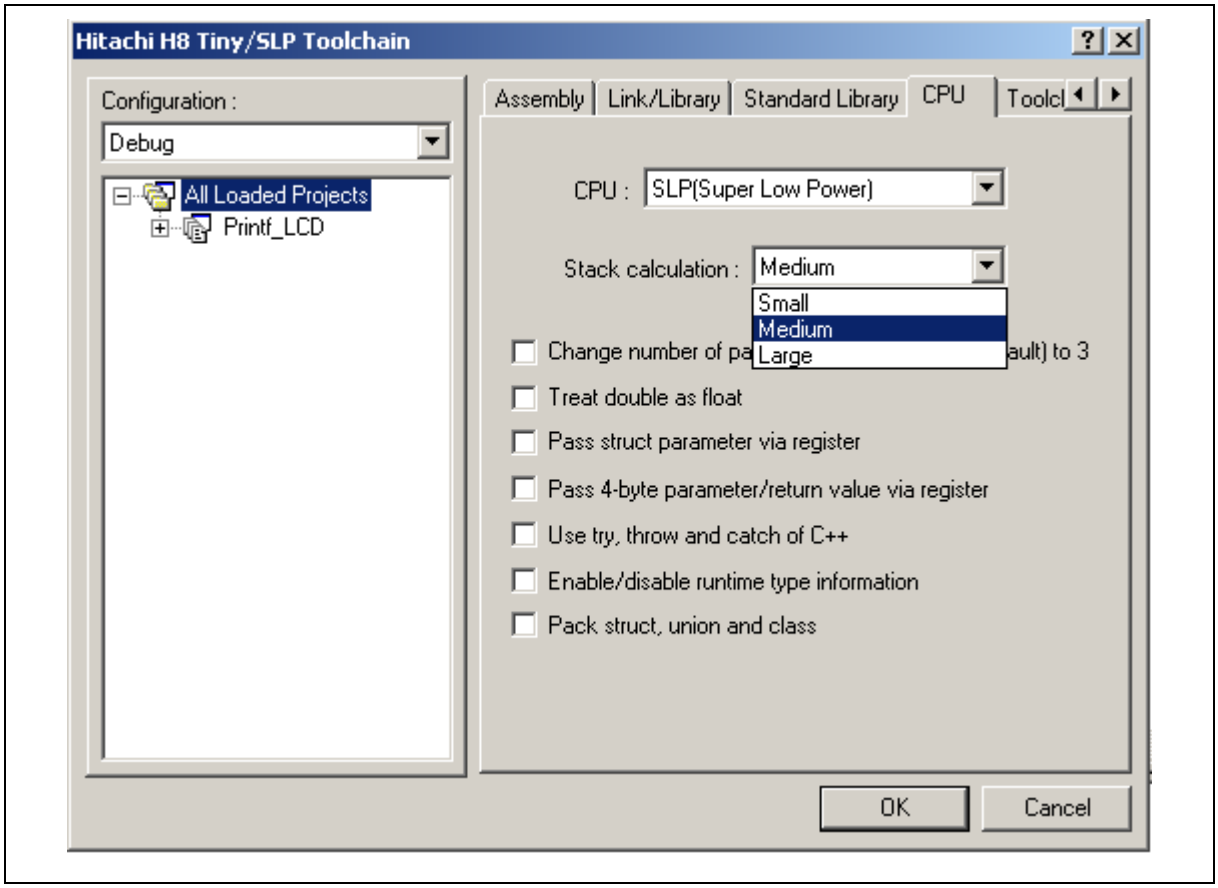

Figure 6 Option Setting (After a project been built)

3.3.1 Operation Mode *(not applicable for this example)*

Operation Mode specifies the operating mode of the object to be created. This option is will only be available depending on the specify CPU series and highly dependant on Address Space (see below). For example, the H8S CPU family with 2357F CPU Type with a 16M Address Space, can only operates in the Advanced mode while Tiny series can only operate in the Normal mode. On the other hand, H8/300H series can operate in either Normal (16 bit addressing) or Advanced mode with 20 or 24 bit addressing.

The HEW project generator will only allow options to be used that are correct for the device chosen.

3.3.2 Address Space *(not applicable for this example)*

Address Space specifies the address space of the object to be created. This option is also device dependant. The HEW project generator will only allow options to be used that are correct for the device chosen.

Please refer to relevant “C/C++ Compiler Assembler Optimizing Linkage Editor User’s Manual” for the list of possible address space.

3.3.3 Merit of Library

The Merit of Library specifies whether the priority of the standard library is speed or code size. One built for small code size, or one built for speed of execution. This can be selected using the drop down list.

If Code Size has been chosen, compiler will reduce code size (number of bytes) and improved code efficiency.

If Speed had been chosen, the time to execute the whole program or function will be minimized. However, this might increase the size of the code.

3.3.4 Stack Calculation

The Stack Calculation affects the way that changes to the value of the stack pointer are calculated by specifying the range of the stack address.
[Small] 1-byte calculation of the stack address.
[Medium] 2-byte calculation of the stack address.
[Large] 4-byte calculation of the stack address.

Small Stack Calculation means that only the lowest byte of the stack pointer is used when adding or subtracting to the value of the stack pointer, this limits the total stack size to H'FE bytes and means that the stack section must start and end on H'100 page boundaries. This reduces the size and speed of instructions used to alter the value of the stack pointer. Because of the limits on the stack size with Small Stack calculations it is not suitable for advanced mode projects or H8/300 / H8/300L projects. This is because the stack will be inadequate for the library required. This setting is only suitable for H8S and H8/300H Normal Mode projects.

Medium Stack Calculation uses only the lowest word of the stack pointer when adding or subtracting to the value of the stack pointer. The gives a total possible stack size of 64k, which is much more than is required for standard H8 applications. The stack section must reside completely within a 64k page of RAM. This setting is the optimum setting for the vast majority of projects.

Large Stack Calculation uses the full four bytes of the stack pointer when adding or subtracting to the value of the stack pointer. This will cause the compiler to operate at all times and places no restrictions on stack size or location in relation to memory page boundaries.

3.3.5 Change the number of parameter registers from 2 (default) to 3

The default for the HEW is to use two registers to pass parameters between functions. This however can be changed to three registers.

If this option is activated, it will change the calling convention used for one function to call another. The user needs to be aware of this when writing assembler functions that are called by, or can call, C functions. As these options are altered the Required C Runtime Library files to be used will change to meet the chosen settings. The usual setting here is to use three registers to pass parameters as this will reduce stack usage and make the code smaller and faster.

3.3.6 Treat double as float

This option forces the compiler to treat all variables declared as type double, as if they were type float. Floats only take up 4 bytes instead of 8 bytes; they do not have the same precision though.

3.3.7 Pass struct parameter via register

This option will force the compiler to pass structure parameters using registers if possible. The standard method for passing structures is by making a copy of the structure on the stack. Structures will only be passed via a register if they are 4 bytes in size or less and so can fit in a register.

3.3.8 Pass 4 byte parameter / return value via register

This option will force the compiler to pass variables of 4 bytes in size using two 16 bit registers (e.g. R0 and R1) instead of passing them via the stack.

3.3.9 Use try, throw and catch of C++

The Use try, throw and catch of C++ option enable C++ exception processing. Enabling this option may reduce the code performance; the default setting is to disable C++ exception processing.

3.3.10 Enable/disable run time type information

The Enable/disable run time type information option enables or disables use of Run Time Type Information for dynamic_cast and typeid. This option is not selected by default and should not be selected for code to be compiled for a library or other relocatable module.

3.4 Setting the Content of Files to be Generated

Click the “Next>” button on the Step-2 screen to display the screen shown in Figure 6.

On this screen, specify information that is necessary to generate files.

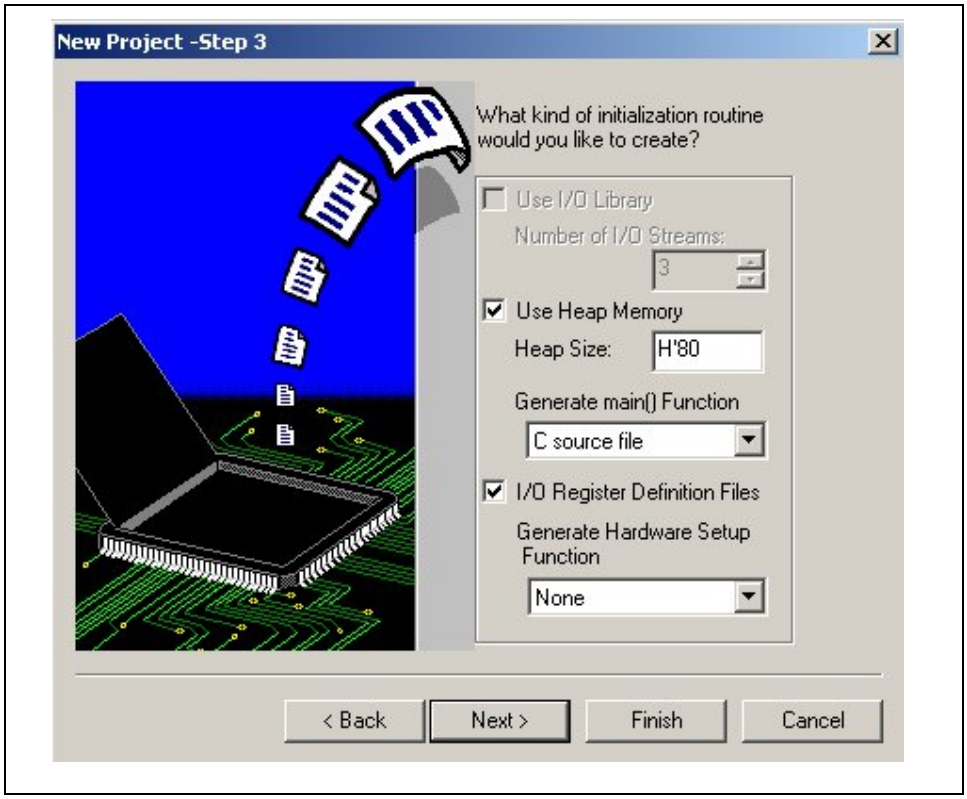

Figure 7 Setting the Content of Files to be Generated (Step 3)

3.4.1 Use I/O Library *(not applicable for this example)*

If Use I/O Library is checked, the standard I/O library can be used. Specify the number of I/O streams to be used simultaneously in the Number of I/O Streams box.

This library is generally for file operation related such as printf, scanf etc.

3.4.2 Use Heap Memory

This is used by functions such as malloc and calloc.

If Use Heap Memory is checked, the function sbrk() for heap area management will be generated. Specify the number of bytes of the heap area in the Heap Size: edit box.

User is advised to uncheck this function.

3.4.3 Generate main() Function

The Generate main() Function option will create a skeleton main function, in a C or C++ source file with the same name as the project. You can also select None from the drop-down list to disable the creation of the main function.

3.4.4 I/O Register Definition Files

The I/O Register Definition Files option will create a header file with definitions for the I/O registers of the on chip peripherals. These can be used in the users code when accessing these registers. This file will be named <iodefine.h> with the content which dependant on device selected.

It is a good idea to generate an iodefine.h header file and take some time to look through it. The <iodefine.h> file uses structures for each peripheral, some of the registers can be accessed using bit fields, so that only the bit or bits required need be considered in the C code, rather that a read modify write of the whole register. This may present opportunities to improve your code, but will involve more modifications to existing projects.

3.4.5 Generate Hardware Setup Function

Then select whether or not to generate a sample code for the program that makes initial settings of the I/O registers from the Generate Hardware Setup Function drop-down list.

The Generate Hardware Setup Function option can be used to generate a skeleton for a Hardware setup function, in a C /C++ source file or Assembly source file, to be run as part of the power up sequence for the device. If selected a setup file is generated which includes inactive code (it is all surrounded by comments) to access every register listed in the <iodefine.h> header file.

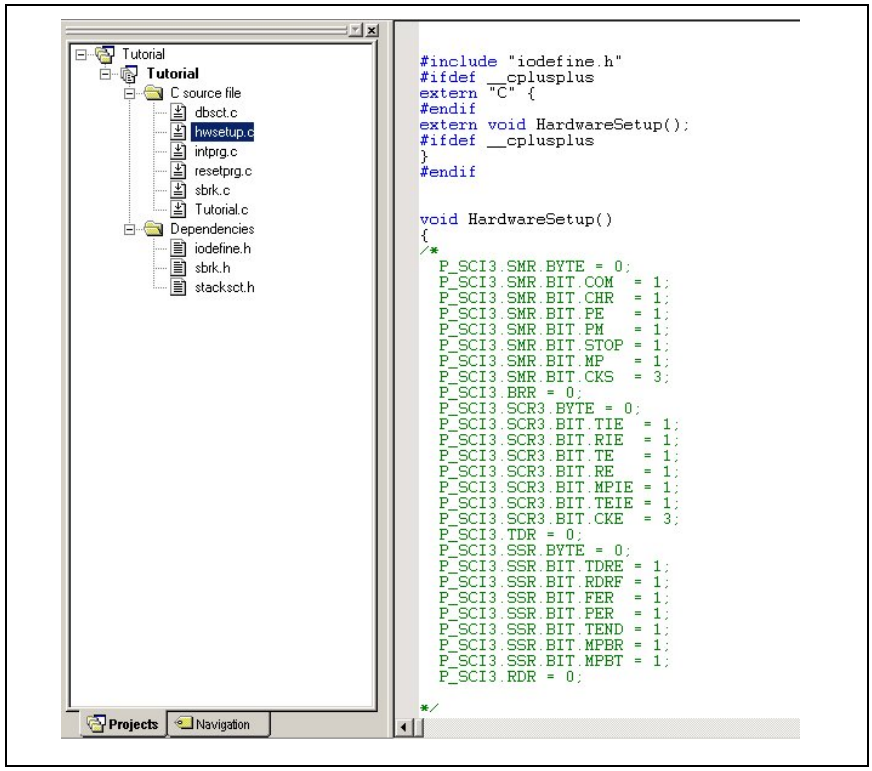

Figure 8 Hardware Setup

Notes: If you want to use an existing main function, uncheck [Generate main() Function]; add the file of the function after generating the project. If the name of the function differs from main, change the caller of the function in resetprg.c.

For the contents of such sample files as a vector table definition file or I/O register definition file that will be generated by the project generator, check the description in the hardware manual for the target CPU.

3.5 Setting the Standard Library

The screen shown in Figure 9 is displayed when the “Next>” button is clicked in the Step-3 screen. This screen is used to set details of compilation by the C/C++ compiler.

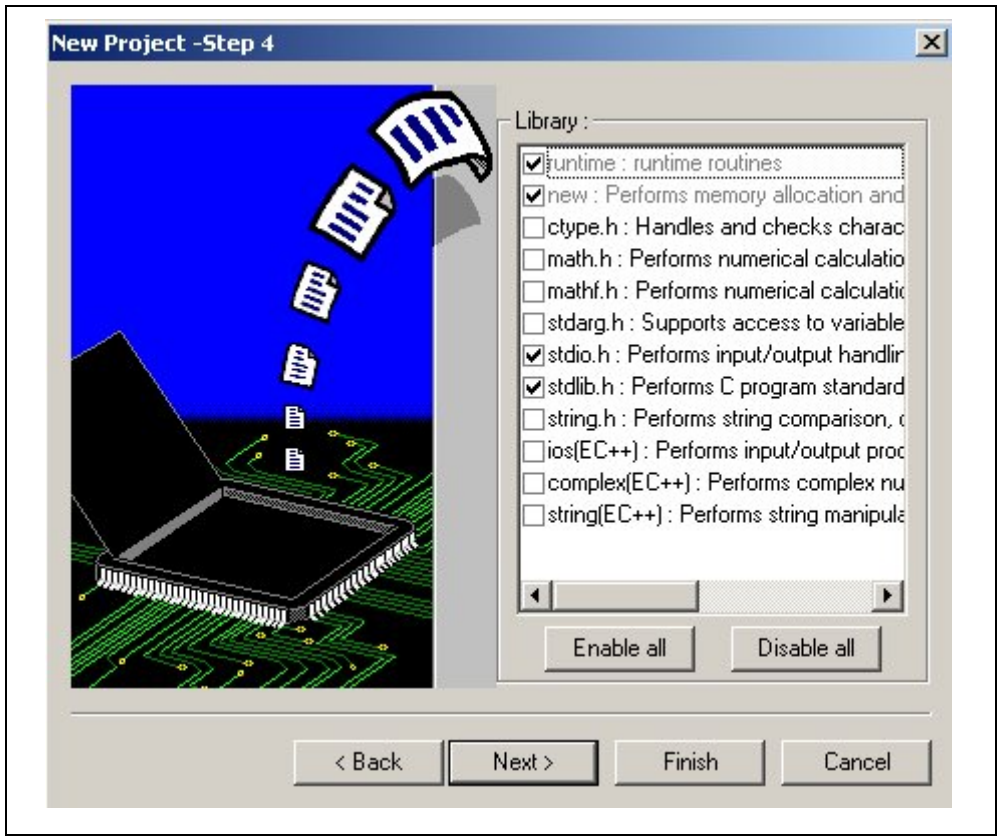

Figure 9 Setting the Standard Library (Step 4)

These settings will set up which modules will be included in the run time library. The HEW compiler includes a library builder utility but pre-built library files are not supplied.

The standard C library modules, which will be used in the project, must be selected to be built by the library generator here. This selection can be altered once the project has been started.

The runtime library module and new library module are not selectable. These modules are selected by default and cannot be unchecked.

Click on “Enable all” to select all standard library functions.

Click on “Disable all” if you do not want to select all standard library functions. For this, only the minimum required functions, runtime and new, are selected.

These header files can also be edited in the C file at later stage.

3.6 Setting the Stack Area

The screen shown in Figure 8 is displayed when the “Next>” button is clicked in the Step-4 screen.

This screen is used to specify the stack area.

The initial value of the stack area differs depending on “CPU Type:” selected in the Step-1 screen. To change the stack size after a project has been created, select the “Project -> Edit Project Configuration” menu item of the HEW window.

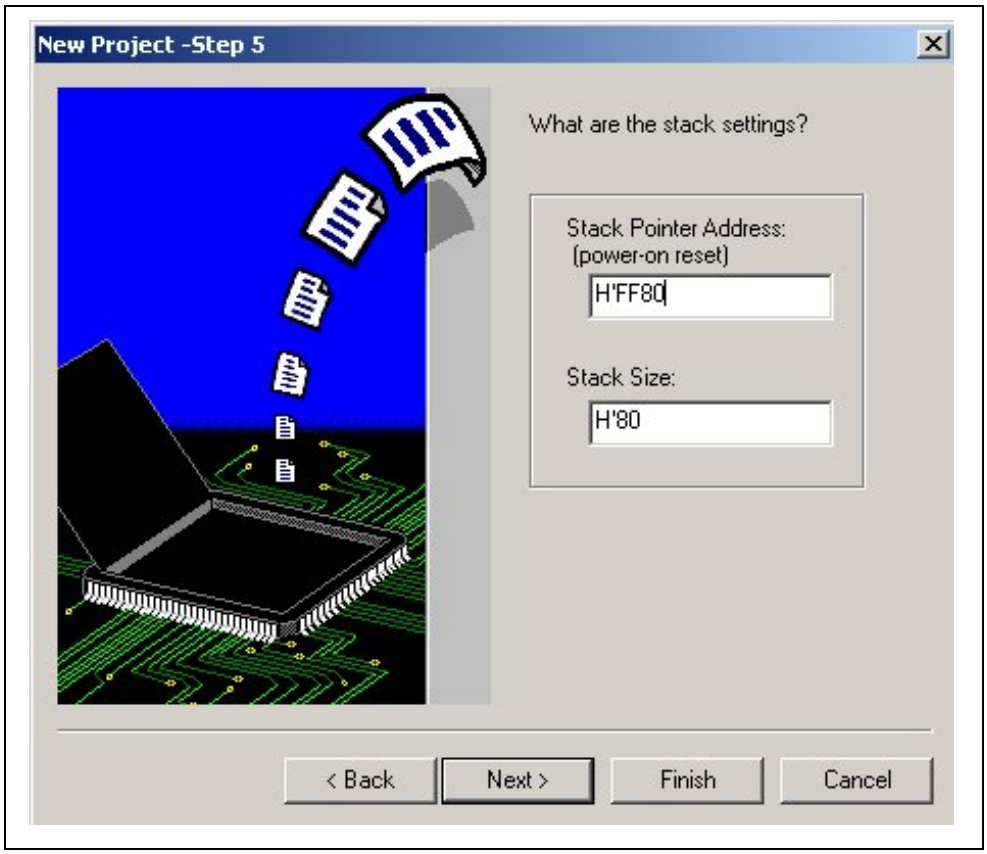

Figure 10 Setting the Stack Area (Step 5)

The stack area is defined in stacksct.h that is generated by the HEW. If stacksct.h has been edited in an editor, its modification after you have selected the “Project -> Edit Project Configuration” menu item of the HEW will not be available. Thus, it is best to set the stack up as required at this stage, although these settings can be altered once the project has been created.

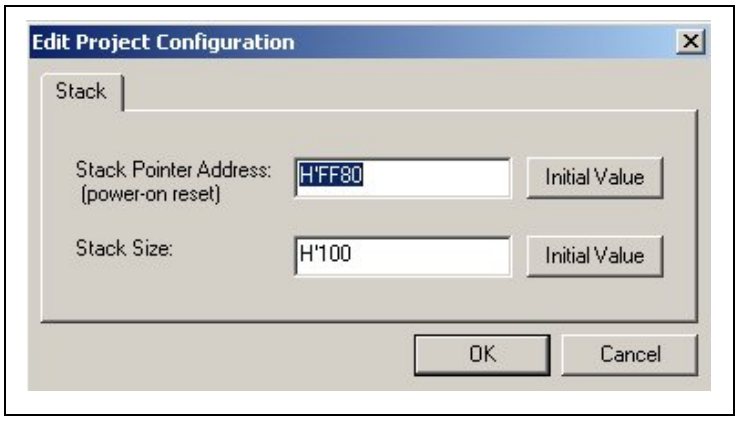

Figure 11 Edit Project Configuration

Note: The naming difference between Start Stack Address and Initial Stack Pointer Address in referring to stack address as below.

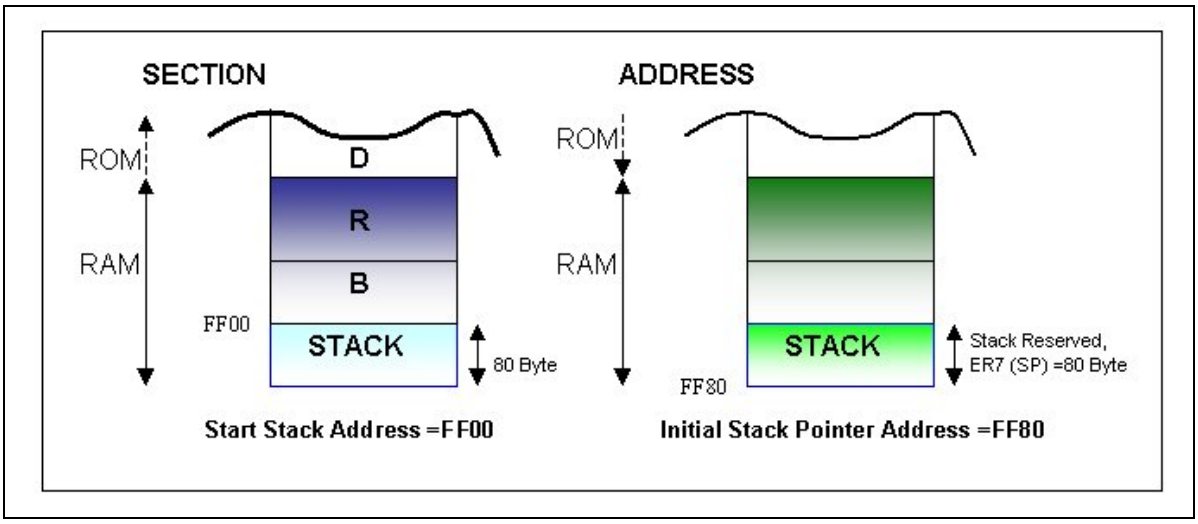

Figure 12 Naming Different Between Start Stack Address & Initial Stack Pointer Address

3.7 Setting the Vector

Clicking the “Next>” button in the Step-5 screen displays the screen shown in Figure 13.

In this screen, a vector is specified.

If “Vector Definition Files” is checked, the HEW creates a vector table definition file. They cover both the reset vectors and the interrupt vectors.

The “Handler” column of “Vector Handlers:” displays the handler program name, and the “Vector” column displays the vector description.

To change the handler program, click the name of the handler program to be changed, and enter a new name.

If the handler name in the “Vector Handlers:” list is changed, the HEW does not create the reset program (resetprg.c).

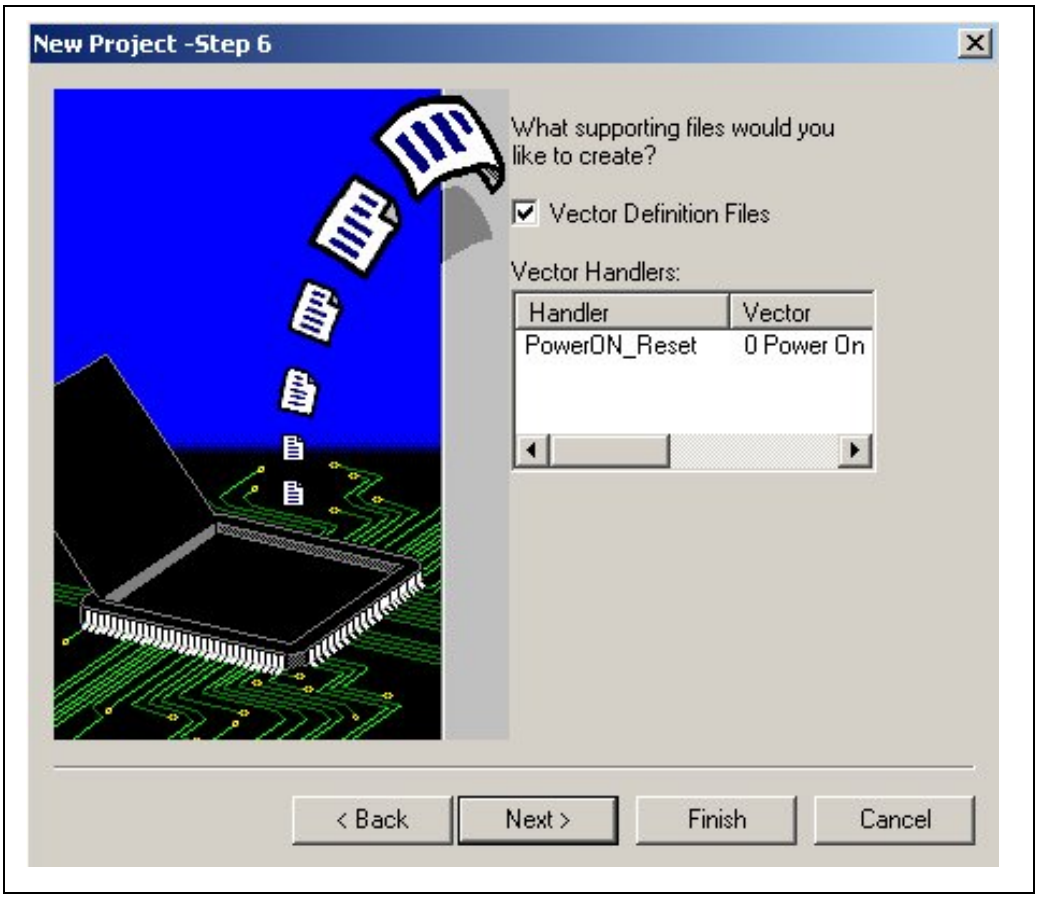

Figure 13 Setting the Vector (Step 6)

3.8 Setting the Target System for Debugging

When the “Next>” button is clicked in the Step-6 screen, the screen shown in Figure 14 is displayed. This screen is used to specify the target system for debugging. Select (check) the target for debugging from the list under “Targets:”. Selection of no target or of multiple targets is allowed.

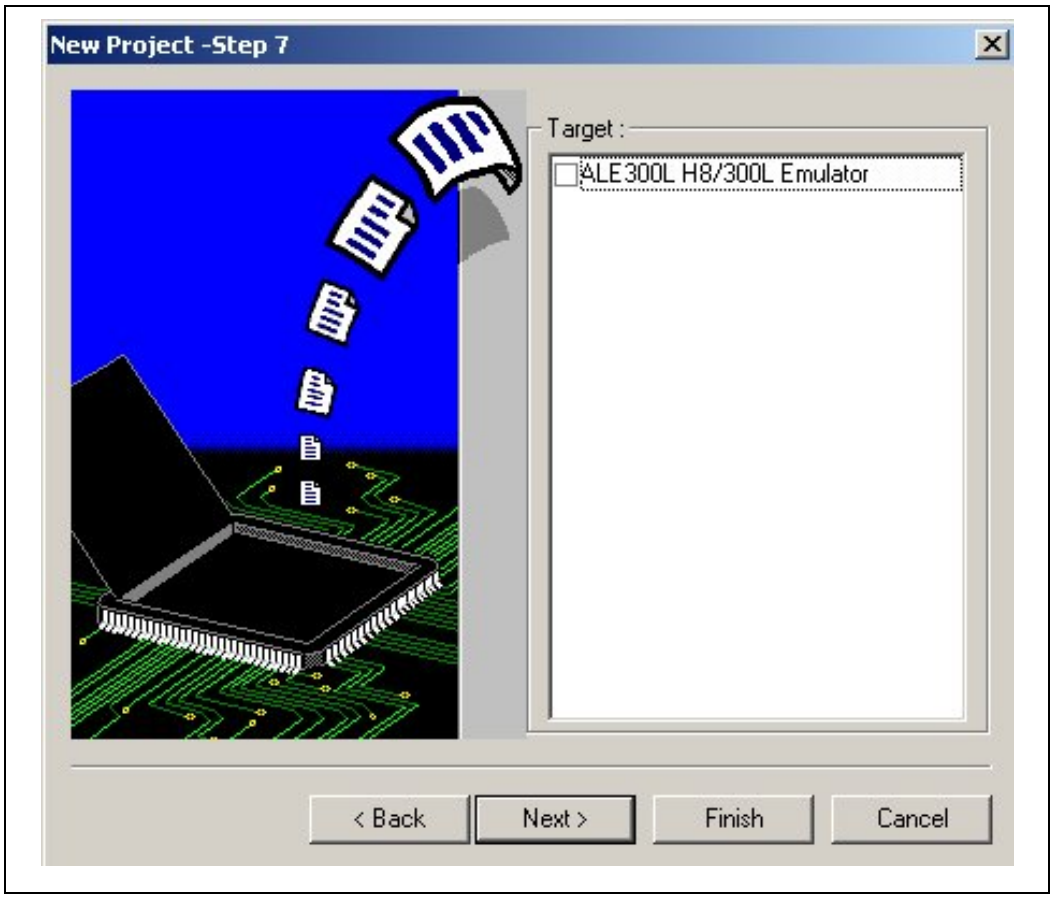

Figure 14 Setting the Target System for Debugging (Step 7)

If you change “Target type:”, you can specify the other target system for debugging.

3.9 Changing the File Name to be Created

When the “Next>” button is clicked in the Step-7 screen, the screen shown in Figure 15 is displayed.

The screen displays a list of files created by HEW according to the previous settings. The [File Name] column in the list shows a file name, “Extension” shows an extension, and “Description” shows the description of a file. The file name can be changed. To change a file name, select it by clicking it and change it to a new file name.

Click the “Finish” button without changing the settings. When the button is clicked, the “Summary” dialog box will be displayed.

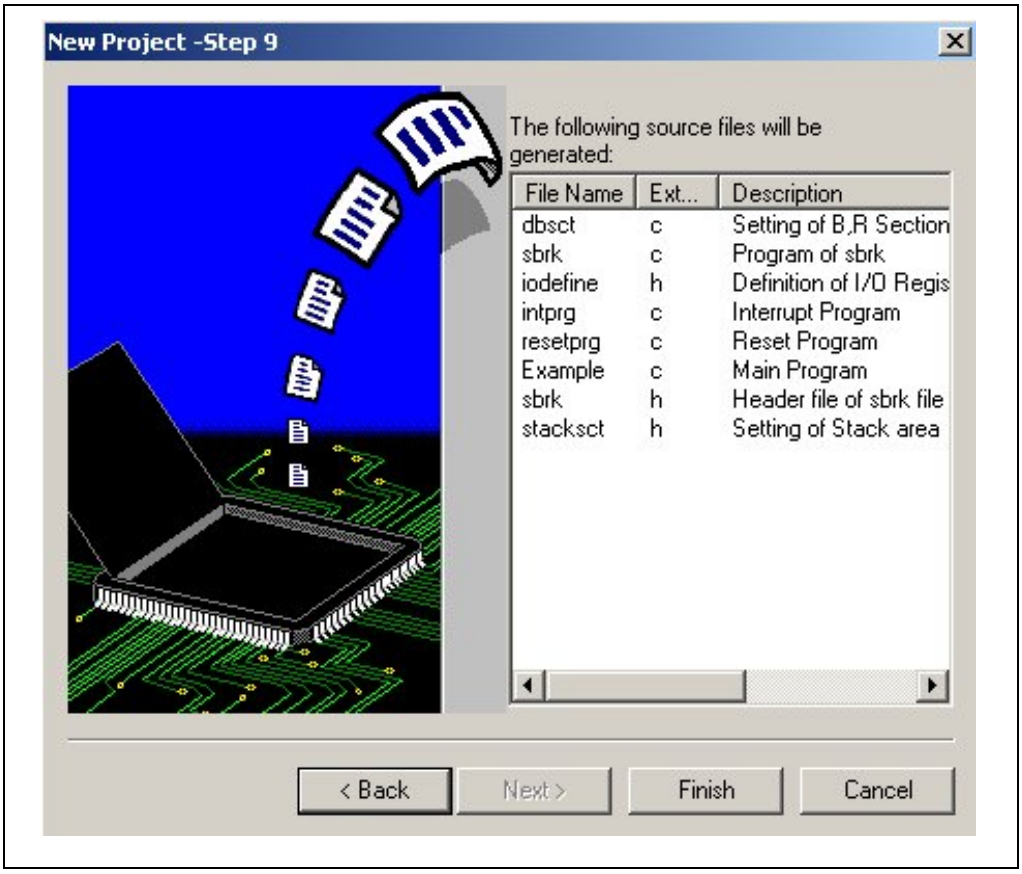

Figure 15 Changing the File Name to be Created (Step 9)

Note: A file with an extension “h” or “inc” (shown in the [Extension] column) is an include file. If you change the file name of an include file, the file name at the include directive have to be modified.

3.10 Confirming Setting (Summary Dialog Box)

When the “Finish” button is clicked, the screen shown in Figure 16 is displayed which shows a list of generated files on the “Summary” dialog box.

Confirm the contents of the dialog box and click “OK”.

When “Generate Readme.txt as a summary file in the project directory” checkbox is checked, the project information displayed on the “Summary” dialog box will be stored in the project directory under the text file name "Readme.txt".

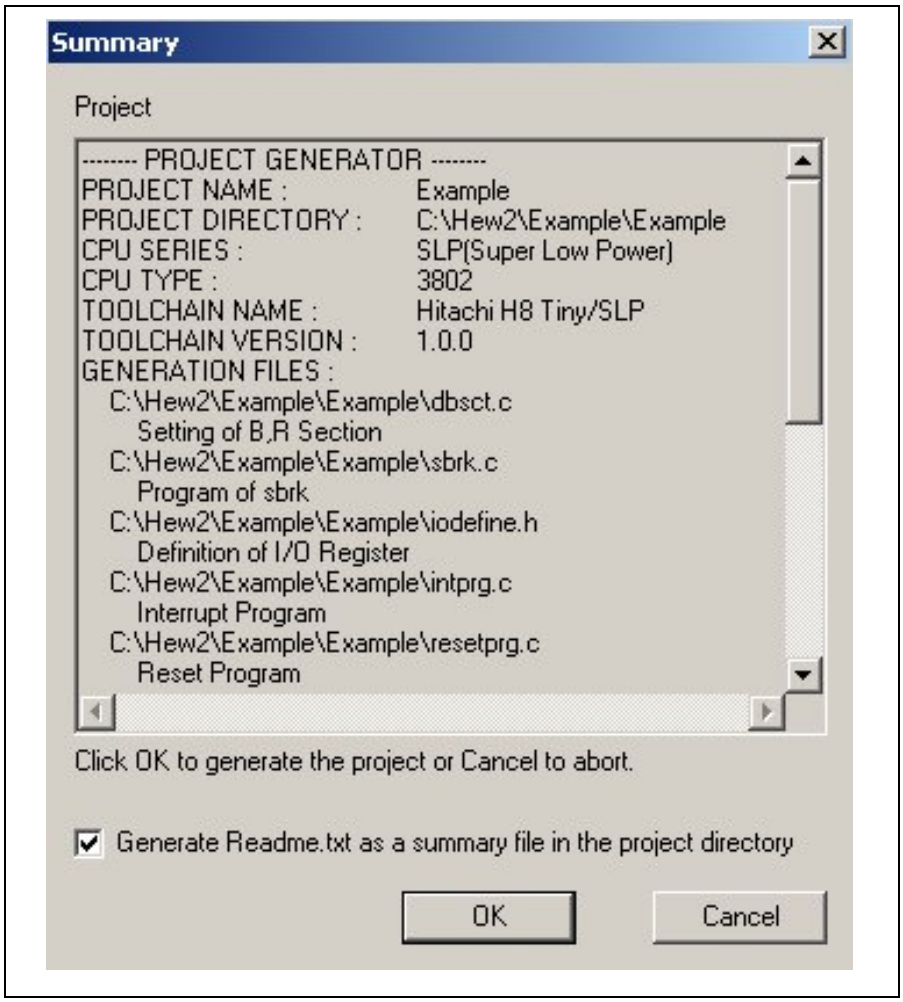

Figure 16 Confirming Setting (Summary Dialog Box)

When the “OK” button is clicked in the Summary Dialog Box, HEW will open a project generated by the project generator (Figure 17).

The project generated by the project generator includes minimum option for the C/C++ compiler, the assembler, the inter-module optimiser and the object converter. Thus, the project can be built.

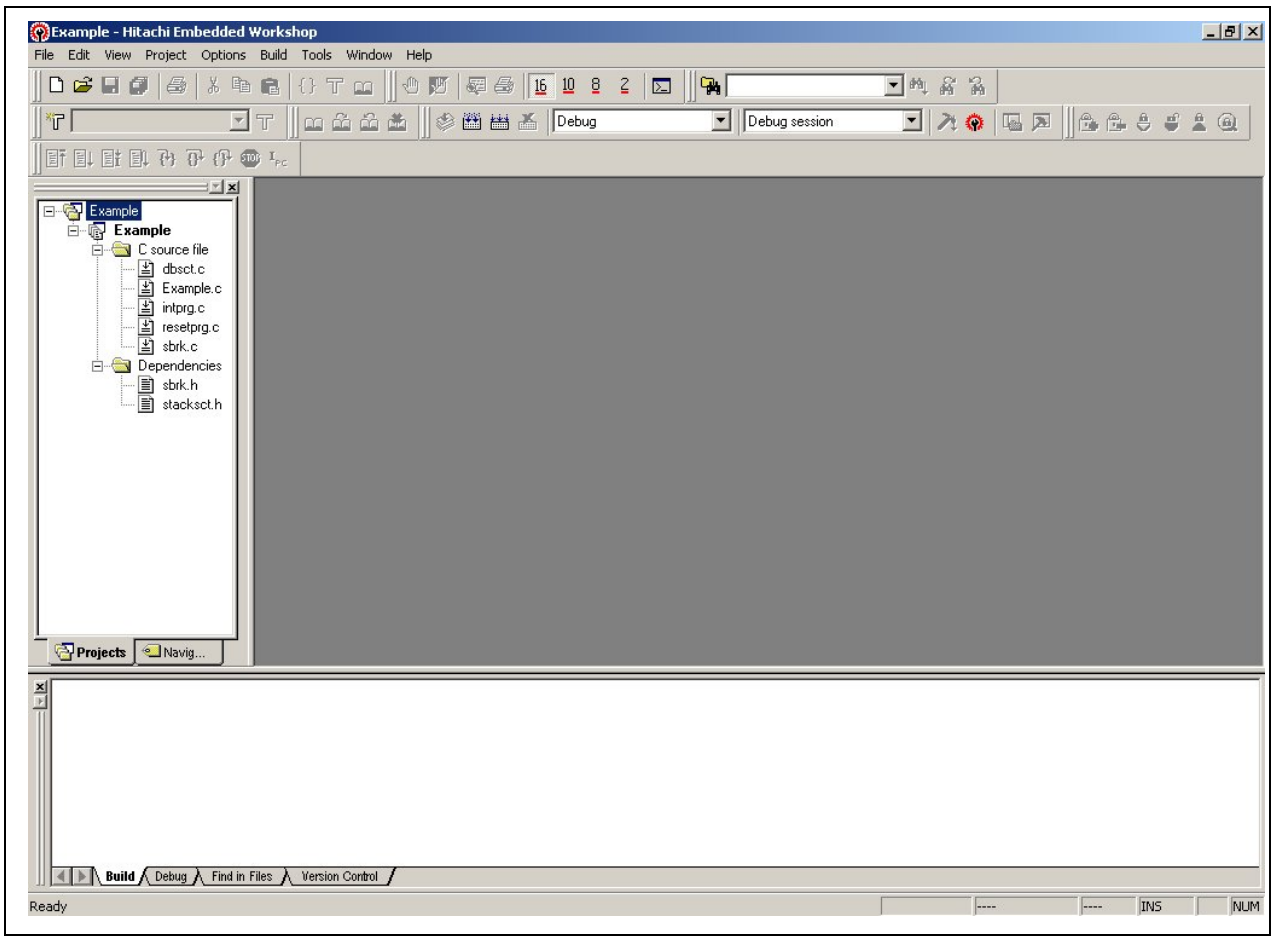

Figure 17 Sub-windows of HEW

For more detail on how to change the state of the HEW and how to use each window such as the editor window, please refer to the “High-performance Embedded Workshop 2 User’s Manual”

4. Files Generated

The following table shows the generated files (with default setting) when using the project generator:

Table 2 Files Generated

"Project type:"	Type of Files Generated *
Application	"dbsct.c"
	"intprg.c"
	"resetprg.c"
	"sbrk.c"
	"<Workspace name>.c"
	"sbrk.h"
	"stacksct.h"
Assembly Application	"intprg.src"
	"resetprg.src"
	"stacksct.src"
	"<Workspace name>.src"
	"vecttbl.src"
	"vect.inc"
Empty Application	No file is to be generated
Library	No file is to be generated

4.1 Project type: -Application

4.1.1 dbsct.c

This file contains the memory map setting for the application. The program will set the starting and end address of ROM and RAM. Besides, it also set the initialize data in ROM (D), initialize data in RAM (R) and also global variable/variable (B).

For more detail about the memory map setting and information, please refer to the respective device Hardware Manual.

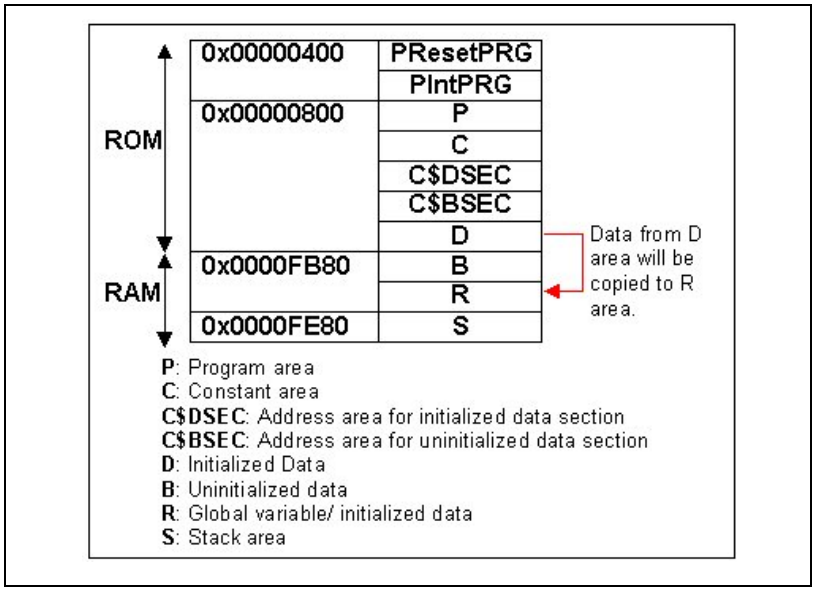

Figure 18 Memory Map

4.1.2 intprg.c

This program contains the entire interrupt vector and interrupt program for the device. In this file, user can write in the interrupt routine for the particular interrupt.

The library file <machine.h> is used to include intrinsic function definitions, e.g. set_vbr. This library header can be removed if none of the intrinsic function will be used.

For example, user can modify or add in the routine for IRQ0 (vector =4) as below.

```
__interrupt(vect=4) void INT_IRQ0(void)
{
    unsigned short delay;
    sound_buzzer(1);
    for (delay = 0 ; delay < 5000 ; delay++);
    sound_buzzer(0);
    clear_irq0_n();
}
```

For more detail information about the interrupt, please refer to the chapter on "Exception Handling" in respective device Hardware Manual.

4.1.3 resetprg.c

This module contains the reset program for the application. The number of exception event is device dependant. For example, for SLP H8/38024, it only has Power On Reset program. For SH3 7727, it has 2 Exception Event, Power On Reset and Manual Reset.

The library file <machine.h> is used to include intrinsic function definitions, e.g. set_vbr. It also can be used in language extensions for embedded systems. This library header can be removed if none of the intrinsic function will be used.

A reset is the highest-priority exception. The internal state of the CPU and the registers of the on-chip peripheral modules are initialized. The reason for the Reset Exception Handling is vary and depend on device.

The few most common reasons are:

- The CPU internal state and the registers of on-chip peripheral modules are initialized.
- The PC is loaded from the reset exception handling vector address, after which the program starts executing from the address indicated in PC.
- When system power is turned on or off.

After the Reset Exception Handling, the application will jump to main() function and execute it.

Besides main(), this module also contains a few pre define function such as Hardware Setup, SIM I/O etc. User can remove the comment on these functions whenever they are not in use.

For more detail information about the interrupt and intrinsic function, please refer to the chapter on "Exception Handling" in the Hardware Manual of respective device.

4.1.4 sbrk.c

This module contains the heap area-setting program. This is used by functions such as malloc and calloc.

The heap segment provides more stable storage of data for a program; memory allocated in the heap remains in existence for the duration of a program. Therefore, global variables (storage class external), and static variables are allocated on the heap. The memory allocated in the heap area, if initialized to zero at program start, remains zero until the program makes use of it. Thus, the heap area need not contain garbage.

4.1.5 <workspace name>.c

This module contains the skeleton of main program for the user to edit.

4.1.6 sbrk.h

This file contains the header file of sbrk file. It defines the heap size according to setting by the user in Step 3 during project generation.

4.1.7 stacksct.h

This file contains the setting of stack area. User should not modify this line.

Select the “Project -> Edit Project Configuration” menu item of the HEW window to change the stack section setting and initial stack value (first instruction)(Figure 11).

```
;-------------------------------------------------------------------------------
;                                                                              |
;   FILE        :resetprg.src                                                  |
;   DATE        :Wed, Jul 09, 2003                                             |
;   DESCRIPTION :Reset Program                                                 |
;   CPU TYPE    :H8/3802                                                       |
;                                                                              |
;   This file is generated by Hitachi Project Generator (Ver.2.1).             |
;                                                                              |
;-------------------------------------------------------------------------------


            .include    "vect.inc"

            .import _main
            .import _PowerON_Reset_SP

CCR_Init:   .equ    B'10000000

            .section    ResetPRG,code
_PowerON_Reset:
_Manual_Reset:
            mov.w   #_PowerON_Reset_SP, SP  ; Initialize SP

            ldc #CCR_Init,CCR

            jsr @_main

            sleep
            .end
```

Figure 19 Initial Stack Pointer

4.2 Project type: -Assembly Application

4.2.1 intprg.src

This file contains the entire interrupt program for the application. The content is similar to intprg.c. Besides the different in the language been used, which is assembly language instead of C/C++, intprg.src does not contain the interrupt vector table.

For detail, please refer to 3.1.2 intprg.c

4.2.2 resetprg.src

This file contains the reset program for the application. The content is the same as resetprg.c but only different in the language been used, which is assembly language instead of C/C++.

For detail, please refer to 3.1.3 resetprg.c

4.2.3 stacksct.src

This file contains the setting of stack area. User should not modify the stack size. The content is similar to stacksct.h. The only different is this file also contain Reset stack pointer.

4.2.4 <workspace name>.src

This module contains the skeleton of main program for the user to edit.

4.2.5 vecttbl.src

This module contains the initialization of the vector table.

For more detail information about the interrupt, please refer to the chapter on "Exception Handling" in the Hardware Manual of respective device.

4.2.6 vect.inc

This module contains the definition of vector table for the application.

For more detail information about the interrupt, please refer to the chapter on "Exception Handling" in respective device Hardware Manual.

5. Start Up Flow

The resetprg.c file prepares the MCU before the actual execution of the application.

A. Initialize the Stack

B. Mask the interrupt

C. Initialize the variable by copying all physical data in the ROM space to the RAM space. For example, a global variable of int count =25 is declared. Variable count had an unknown data in RAM when power up. The INITSTC.c routine will copy the initial data “25” from the ROM space (D are) to the RAM space (R area).

D. Initialize all the necessary I/O function if it’s enable in Step 3.

E. Setup all peripherals (hardwareSetup.c).

F. Clear internal mask.

G. Jump to main routine where by the actual application begin.

```
//#ifdef __cplusplus            // Remove the comment when you use global class object
//extern "C" {                  // Sections C$INIT and C$END will be generated
//#endif
//extern void _CALL_INIT(void);
//extern void _CALL_END(void);
//#ifdef __cplusplus
//}
//#endif

#pragma section ResetPRG (i)

__entry(vect=0) void PowerON_Reset(void)
{
     set_imask_ccr(1);(ii)
    _INITSCT();(iii)

//  _CALL_INIT();                 // Remove the comment when you use global class object

//  _INIT_IOLIB();     (iv)       // Remove the comment when you use SIM I/O

//  errno=0;                      // Remove the comment when you use errno
//  srand(1);                     // Remove the comment when you use rand()
//  _s1ptr=NULL;                  // Remove the comment when you use strtok()

    HardwareSetup(); (v)        // Use Hardware Setup
    set_imask_ccr(0);(vi)

    main(); (vii)

//  _CLOSEALL();                  // Remove the comment when you use SIM I/O

//  _CALL_END();                  // Remove the comment when you use global class object

    sleep();
}

//__interrupt(vect=1) void Manual_Reset(void)   // Remove the comment when you use Manual Reset
//{
//}
```

Figure 20 Start Up Flow

6. Porting from A Device to Another Device

HEW provides a flexible code development and debugging environment for applications targeted at Renesas microcontrollers. Thus, it is easy to port a program from one device to another device. In the following example, we quoted the sample program from *Interfacing to E^2PROM with I^2C Emulation (Port)* to demonstrate the simplicity of porting the program from SLP H8/38024 (Workspace name: I2C) to Tiny 3644 (Workspace name: 3644).

For full detail and explanation on how the program works, please refer to Application Note Interfacing to E^2PROM with I^2C Emulation (Port).

6.1 Checking the Available Files

In this step, we will need to check and confirm which files were generated by the HEW. For those files generated by the HEW, user need to check whether any modification been made.

Please refer to 3.0 Files Generated for detail.

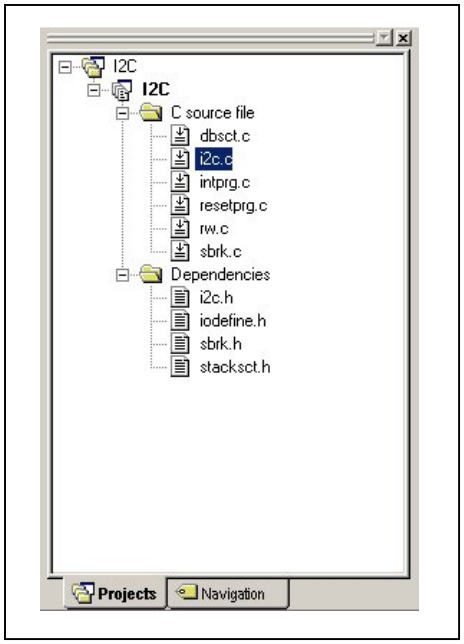

Figure 21 Files in I2C Workspace

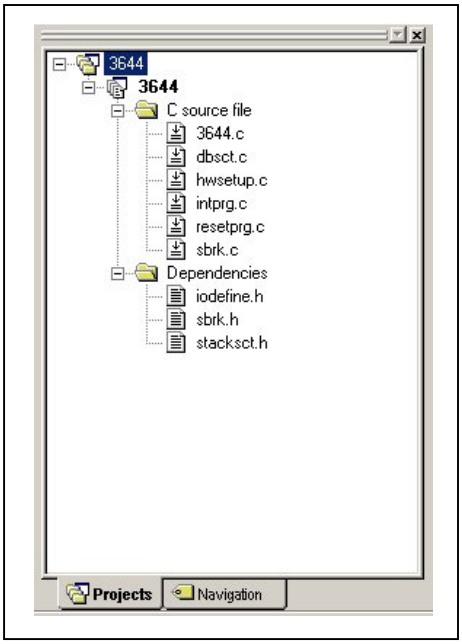

Figure 22 Initial Files in 3644 Workspace

In this example, as the project generator will create all necessary setting for the new device; we realized that most of the files were auto generated except “rw.c” and “i2c.h”.

Note: Users are advice to generate the project target to the new device (3644) and port the code from other project into the new project.

6.2 Copying and Adding the Files

Copy the content of “rw.c” and “i2c.h” file from I2C Workspace to 3644 workspace. Save both files according to its original name. Then add the files into 3644 workspace.

Not to forget “i2c.c” which contains the main program also been edited.

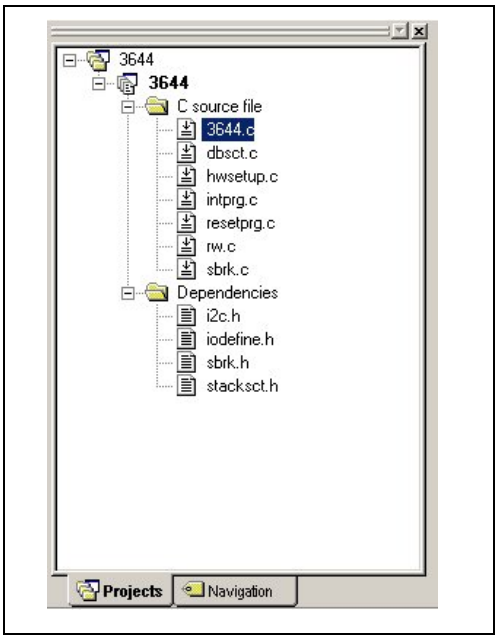

Figure 23 Files in 3644 Workspace After Added “rw.c” & “i2c.h”

Note: If you try to compile or build the application in 3644 workspace, error message will be prompted out. This is because further modification is needed before its error free.

For more information on modifying the project, please refer to “Chapter 3 Modifying the Project” in “High-performance Embedded Workshop 2 Tutorial” manual.

6.3 Understanding the Code

After understanding the code, you will need to locate the critical code line, which is device dependant such as IO pin configuration and specific register configuration.

For example, Figure 24 below shows the definition of Tiny 3644 pin configuration in “i2c.h” file (I2C workspace). There is no specific register configuration been used in this example.

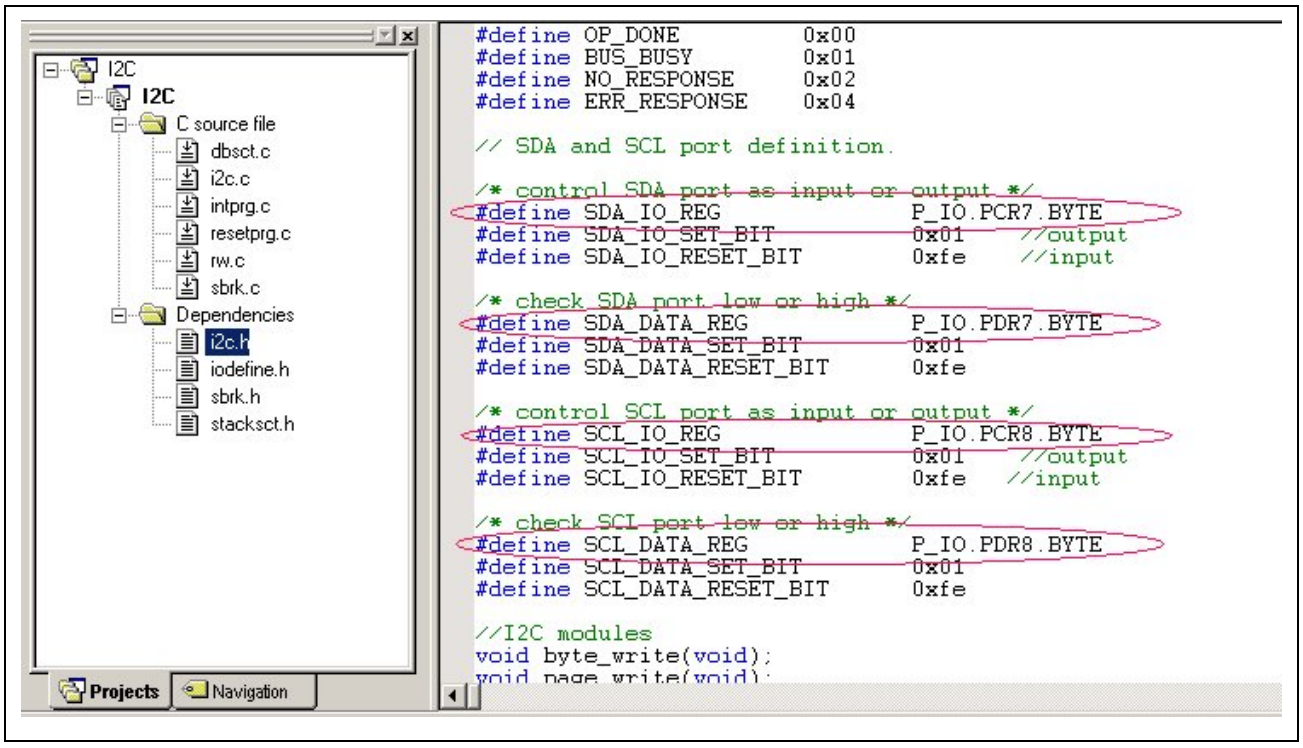

Figure 24 “i2c.h” file of I2C Workspace (SLP H8/38024)

From the earlier explanation, we understand that most of the generated files are generic across all kind of device except <iodefine.h>(contains the entire pins definition), which is device dependant.

Thus, when porting this program to 3644 device, these definitions need to be modified.

6.4 Modifying the Program

Before modifying the code, we need to compare the <iodefine.h> file for both devices.

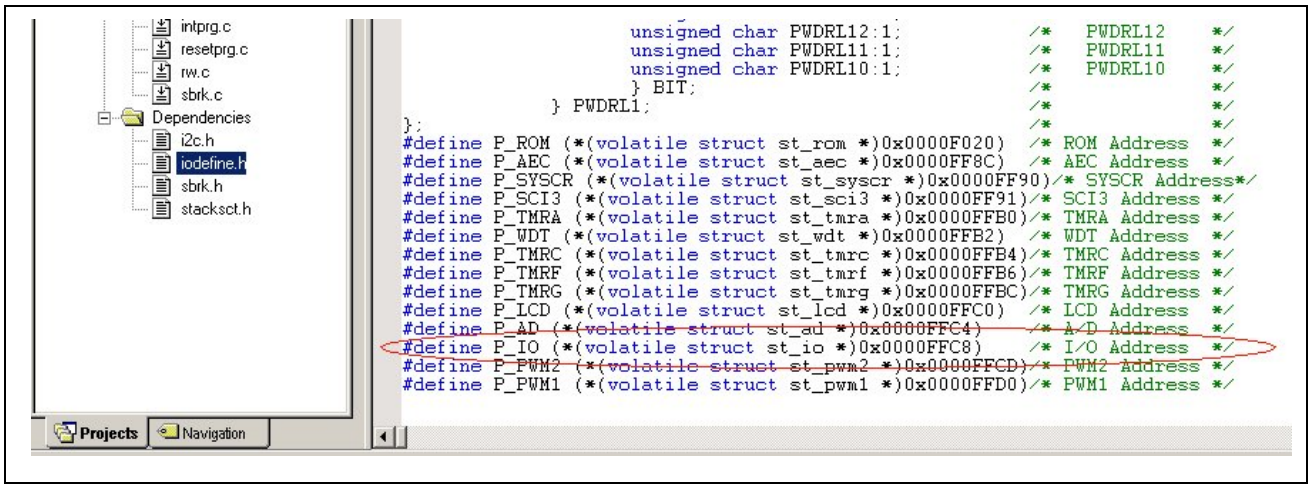

Figure 25 I/O address Definition of I2C Workspace (SLP H8/38024)

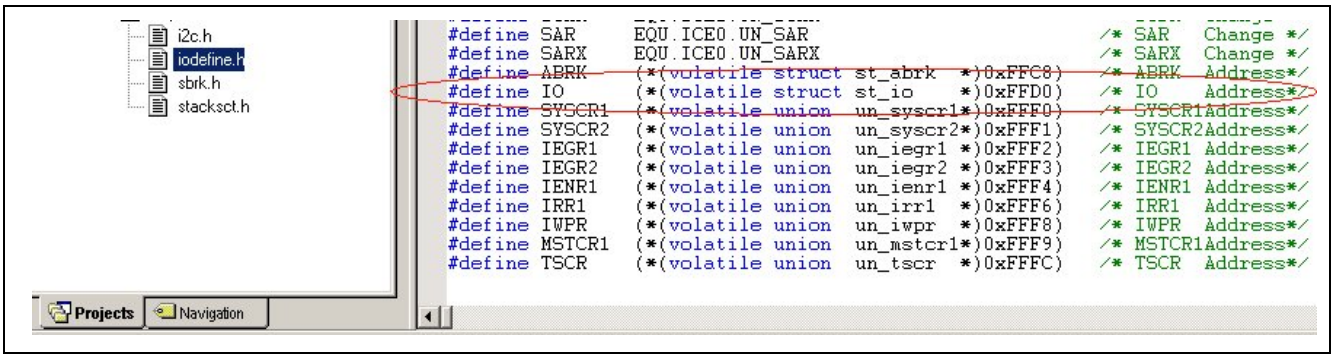

Figure 26 I/O address Definition of 3644 Workspace (Tiny 3644)

From the <iodefine.h> file, we can see that both devices have general IO port. However, the defined name for both IO port are different (Figure 25 & Figure 26). Thus, after porting the code to 3644, we need to modify “P_IO.PDR7.BYTE” to “IO.PDR7.BYTE”. Same modification also needs to be done on other similar definitions.

Besides the above comparison, we also realized that the bit definitions of PDR7 and PDR8 for both devices are the same. Thus, modification is not needed (Figure 27 & Figure 28).

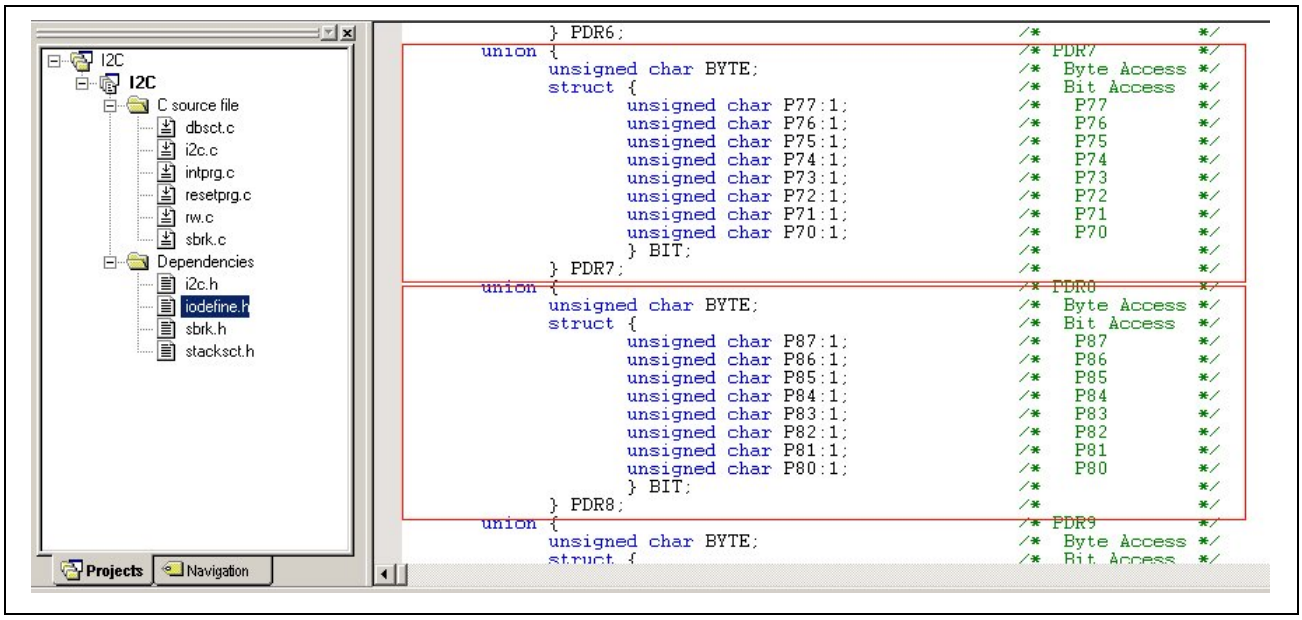

Figure 27 PDR7 & PDR8 Definition of I2C Workspace (SLP H8/38024)

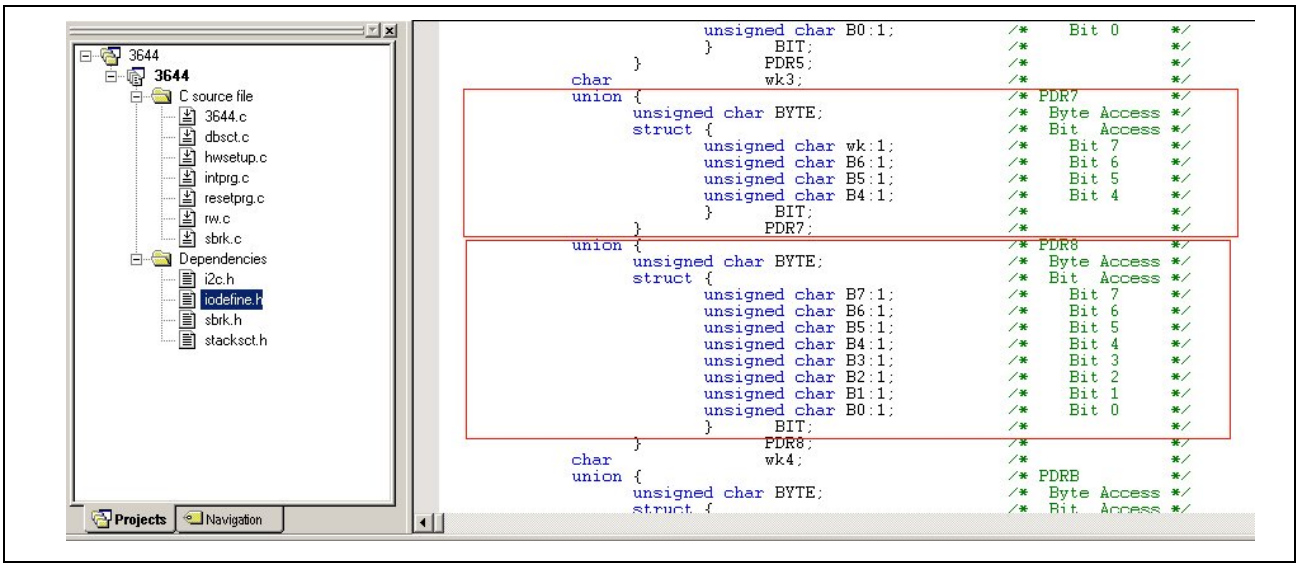

Figure 28 PDR7 & PDR8 Definition of 3644 Workspace (Tiny 3644)

The same comparison is carried out for PCR7 and PCR8. There is a different in definition for both devices (Figure 29 & Figure 30). Thus, modification is needed for PCR7 and PCR8. We need to change “P_IO.PCR8.BYTE” to “IO.PCR8” in 3644 workspace.

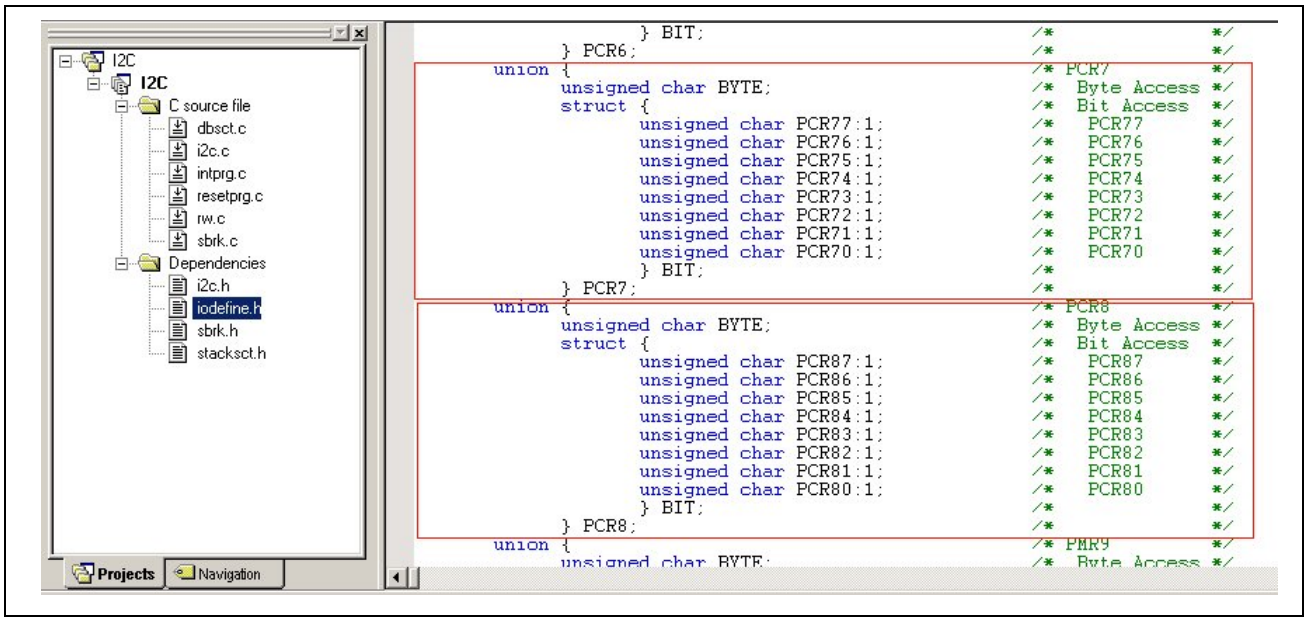

Figure 29 PCR7 & PCR8 Definition of I2C Workspace (SLP H8/38024)

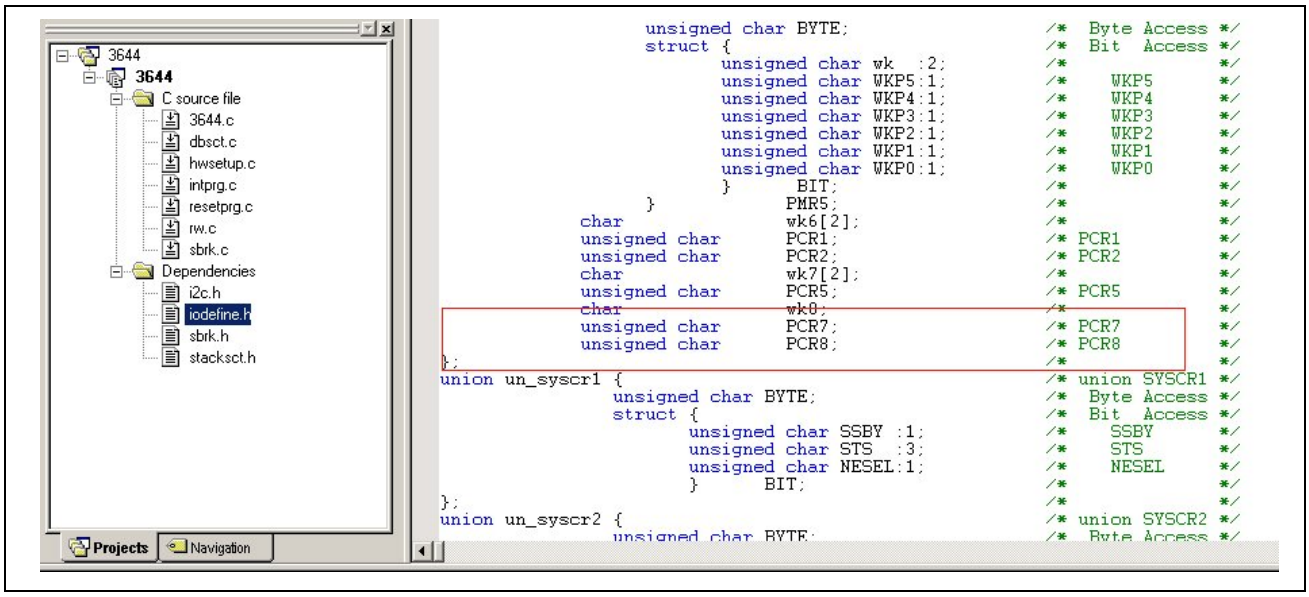

Figure 30 PCR7 & PCR8 Definition of 3644 Workspace (Tiny 3644)

After modification, "i2c.h" file in 3644 workspace as shown in Figure 31.

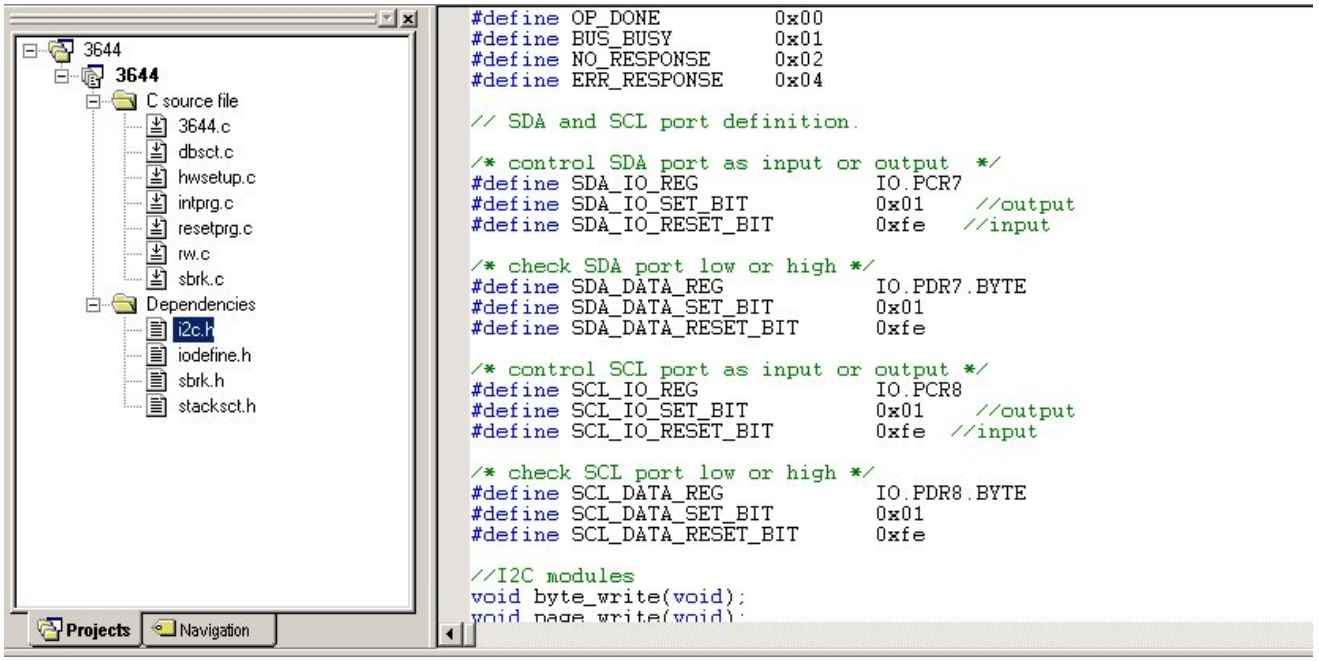

Figure 31 "i2c.h" file of 3644 Workspace (Tiny 3644) after modification

Compile and build the workspace and Tiny 3644 is ready to interface with E^2PROM using I^2C Emulation (Port).

The above example is to demonstrate the wonders of C portability. It is a basic example of how to deal with the C files.

However, in a more complex and microcontroller with different features and architecture, the porting will be much tougher,

Considerations such as:-

1. Memory space, example small RAM size
2. Peripheral channels such as 2 serial ports but only one is available peripheral usage, timer etc,
3. Different compiler with different directive, and
4. others

are important before doing the porting.

REFERENCE

1. *Application Note "The HEW Project Generator For The Hitachi H8 v4 compiler" (Issue: APP20011101-01)*, Hitachi Micro Systems Europe Ltd, 16.11.2001.

2. "*SuperH RISC Engine Hitachi Embedded Workshop 2.0 Tutorial (Hitachi Toolchain)*", Hitachi Ltd, 16.10.2001.

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	September.03	—	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.